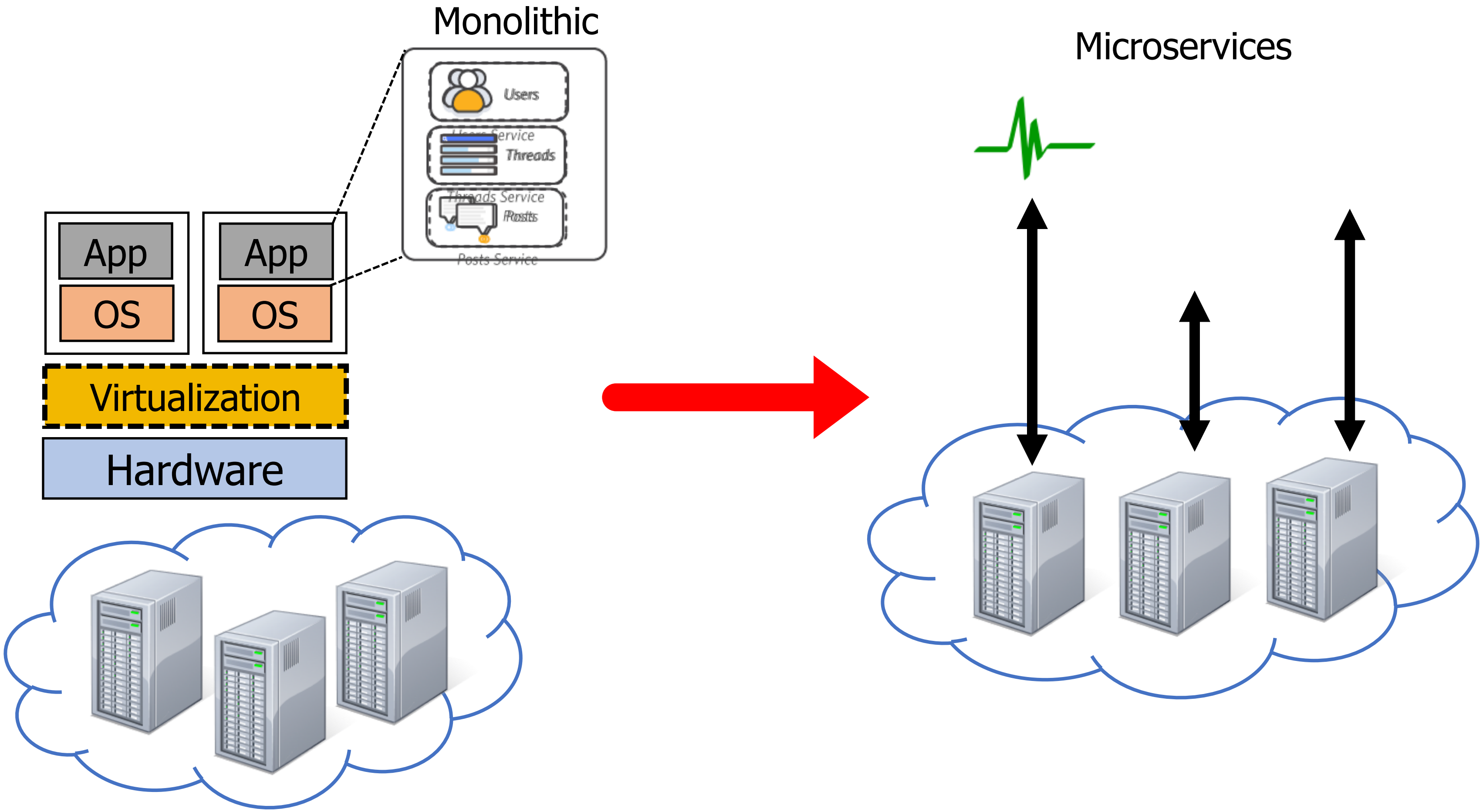
GrandSLAm: Guaranteeing SLAs for Jobs in Microservices Execution Frameworks

Ram Srivatsa Kannan, Lavanya Subramanian, Ashwin Raju,
Jeongseob Ahn, Jason Mars, Lingjia Tang

Transformation of Cloud Services



Monolithic

Users

Users Service

Threads

Threads Service

Posts

Posts Service

App

OS

App

OS

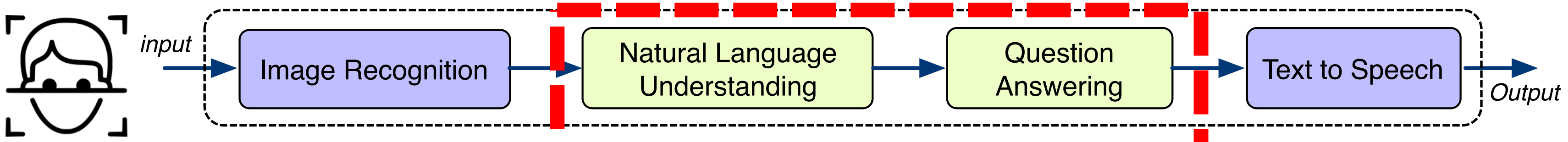
Virtualization

Hardware

Microservices

Building Applications with Microservices

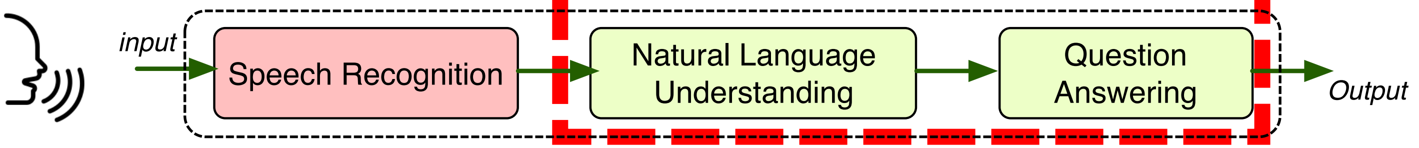
Duplicated microservices



App: Image query (4 microservices)

- Recognizes the input image
- Generates natural language descriptions of the images
- Builds a sentence for the description
- Outputs the sentence as voice

Low Resource Utilization

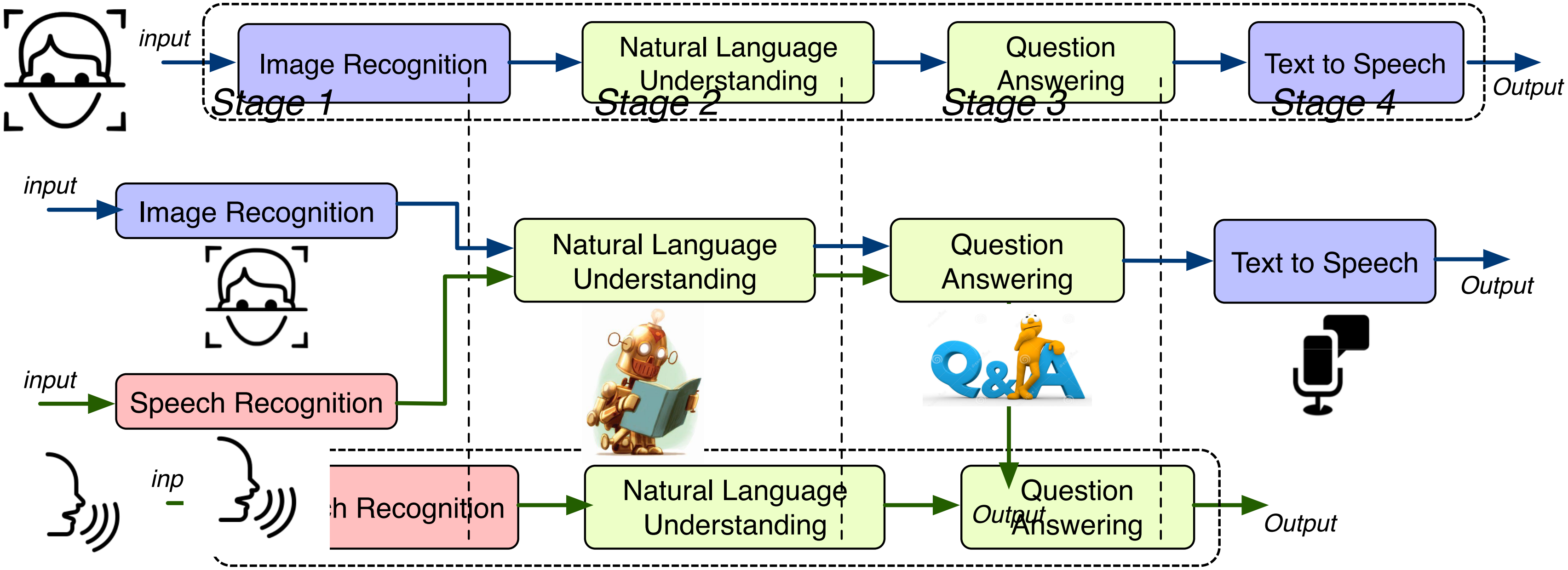


App: Intelligent personal assistant (3 microservices)

- Provides answers to queries that are given as input through voice

Sharing Microservices

- Amalgamate redundant microservices



Sharing microservices can improve resource utilization

How does instance sharing actually happen?

Impact on resource utilization?

Approach in AI & ML Microservices

- Batching multiple requests[1]

- Requests belonging to the different applications can be composed into a single batch

App A

App B

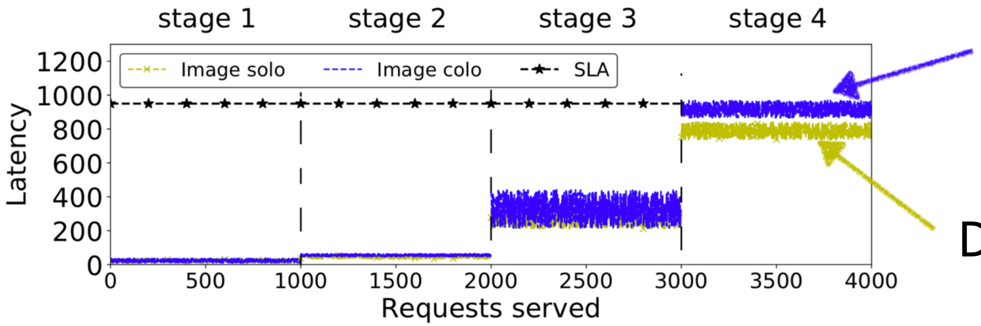
App C

Sharing degree (batch size): 3

[1] Djinn and Tonic: DNN as a Service and Its Implications for Future Warehouse Scale Computers, ISCA 15

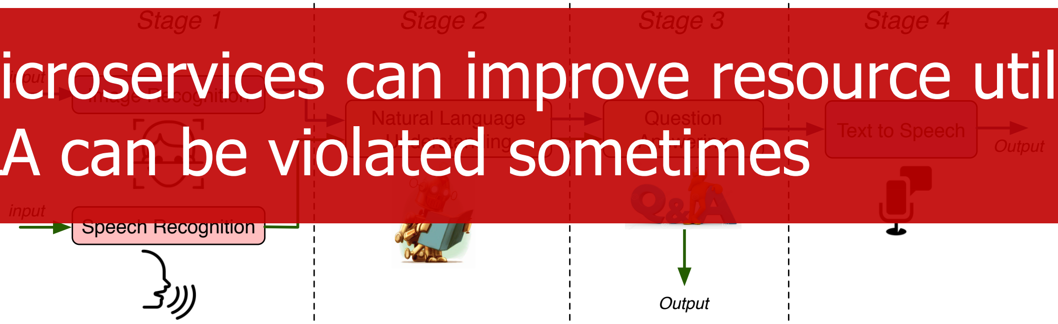
Impact of Sharing Microservices

Image query
(4 microservices)



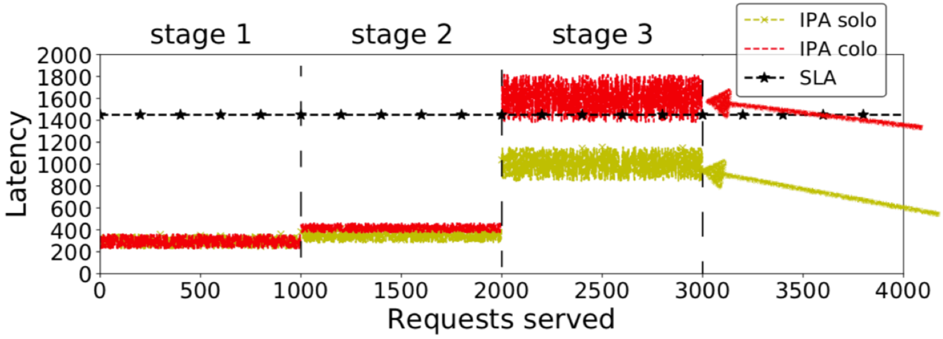
Allow sharing

Disallow sharing



Sharing microservices can improve resource utilization, but the SLA can be violated sometimes

Intelligent personal assistant
(3 microservices)

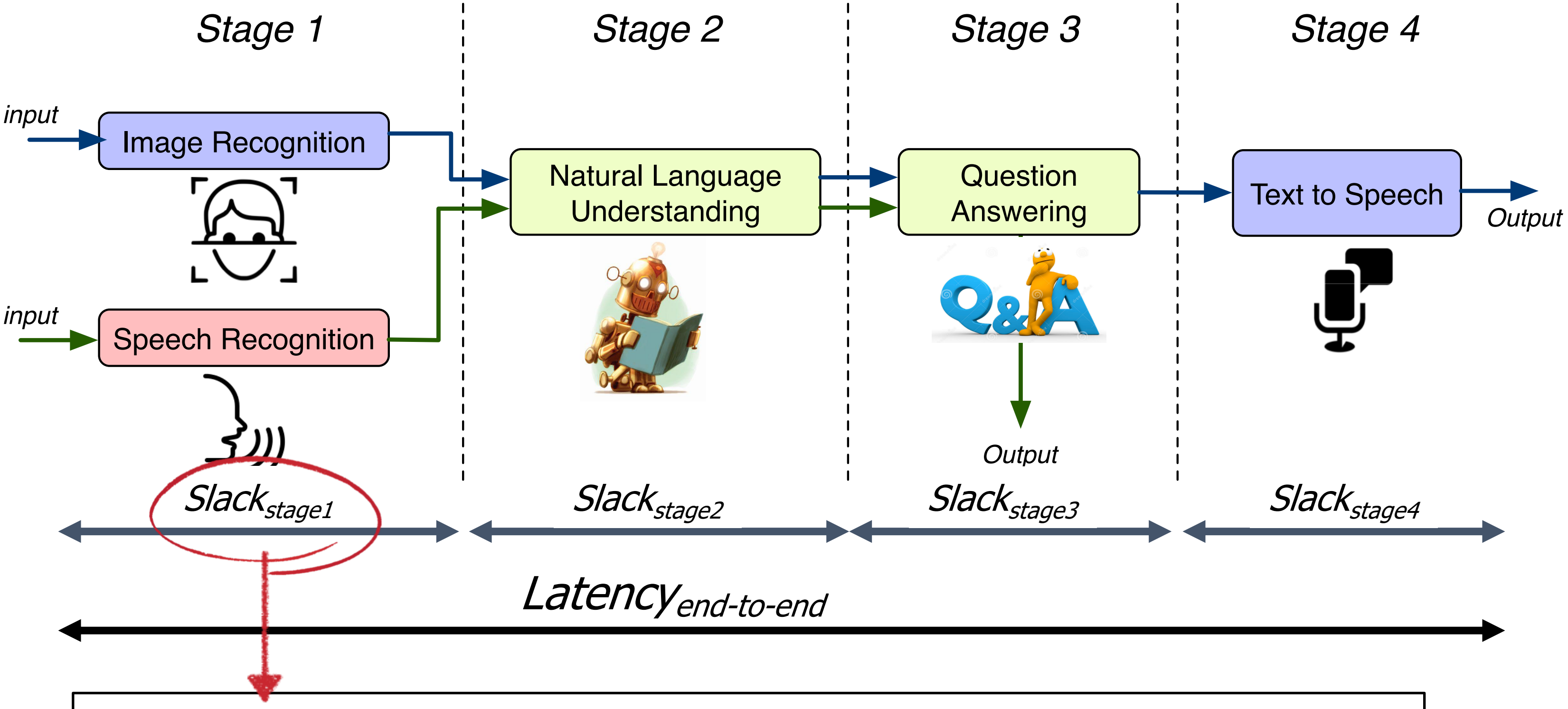


Allow sharing

Disallow sharing

Latency Aware Sharing – Holy Grail of Multi-tenancy in Microservices

- What is a necessary condition?



Stage 1

Stage 2

Stage 3

Stage 4

input

Image Recognition

input

Speech Recognition

Natural Language Understanding

Question Answering

Output

Text to Speech

Output

$Slack_{stage1}$

$Slack_{stage2}$

$Slack_{stage3}$

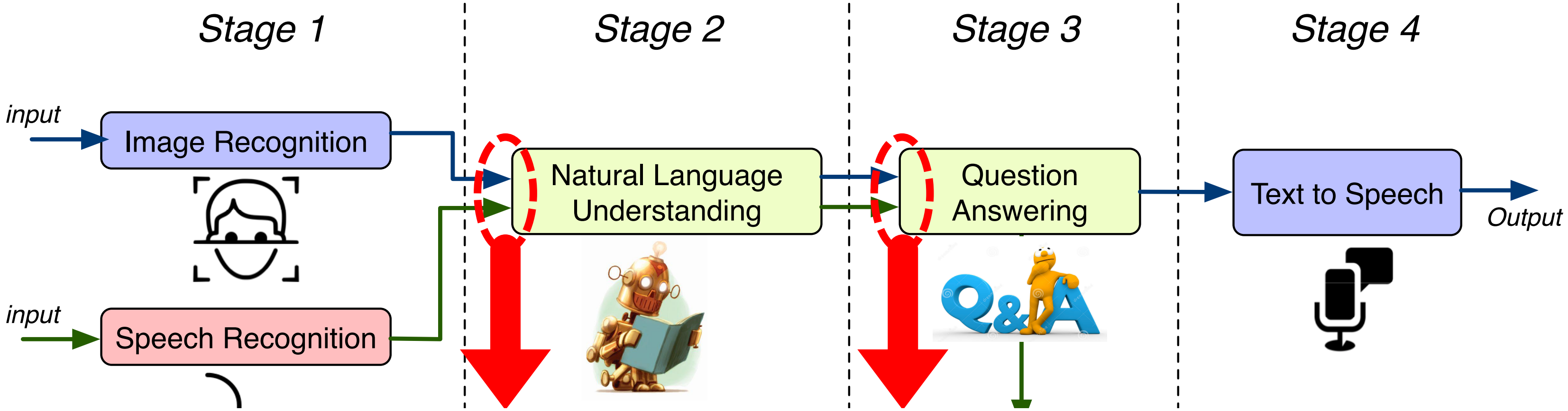
$Slack_{stage4}$

$Latency_{end\text{-}to\text{-}end}$

The maximum amount of time, a request can spend at the stage

Enabling Sharing Microservices

- What is a necessary condition?

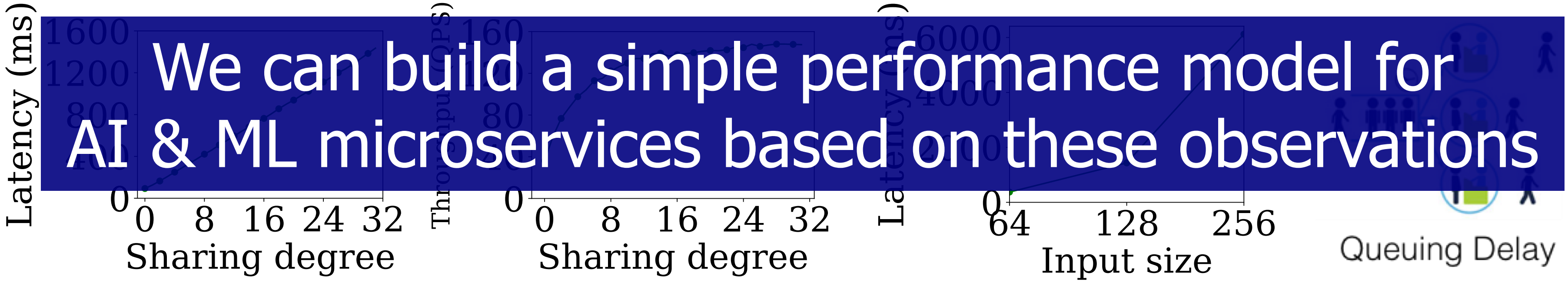


Goal 1: Accurately estimate completion time for any given request.

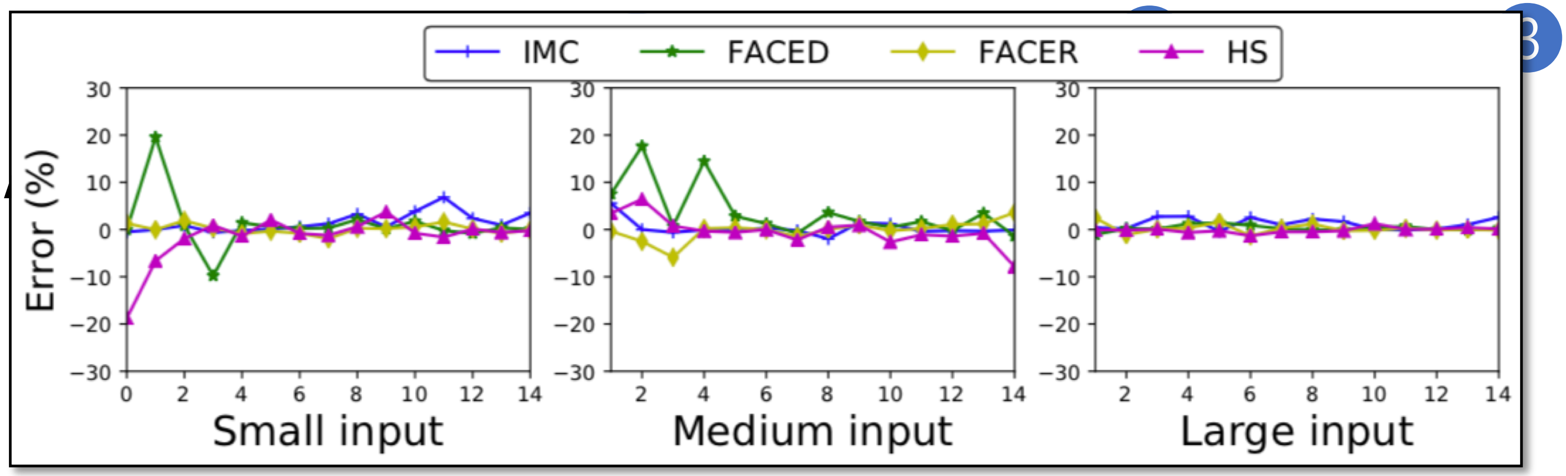
Goal 2: Identify slack at each microservice stage.

Towards Predicting The Execution Time

- Performance study: image recognition

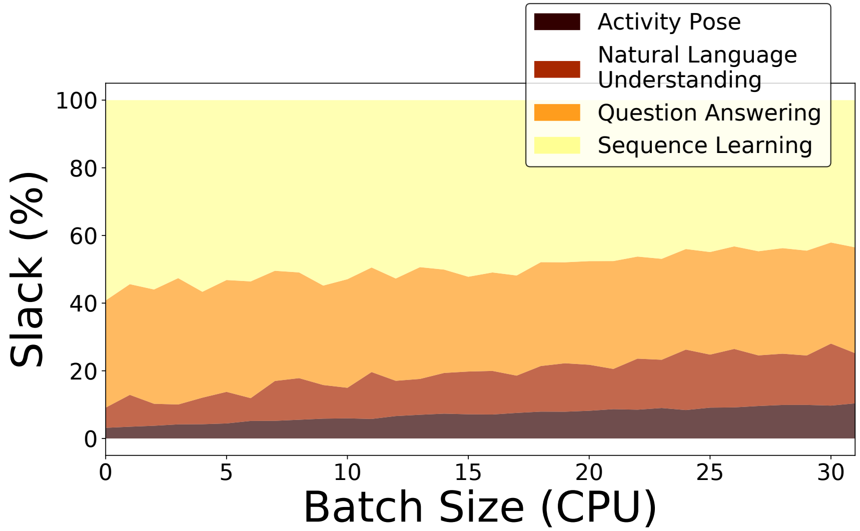


We can build a simple performance model for AI & ML microservices based on these observations



Calculating Microservice Stage Slack

- Stage slacks are proportionally allocated from the end-to-end latency



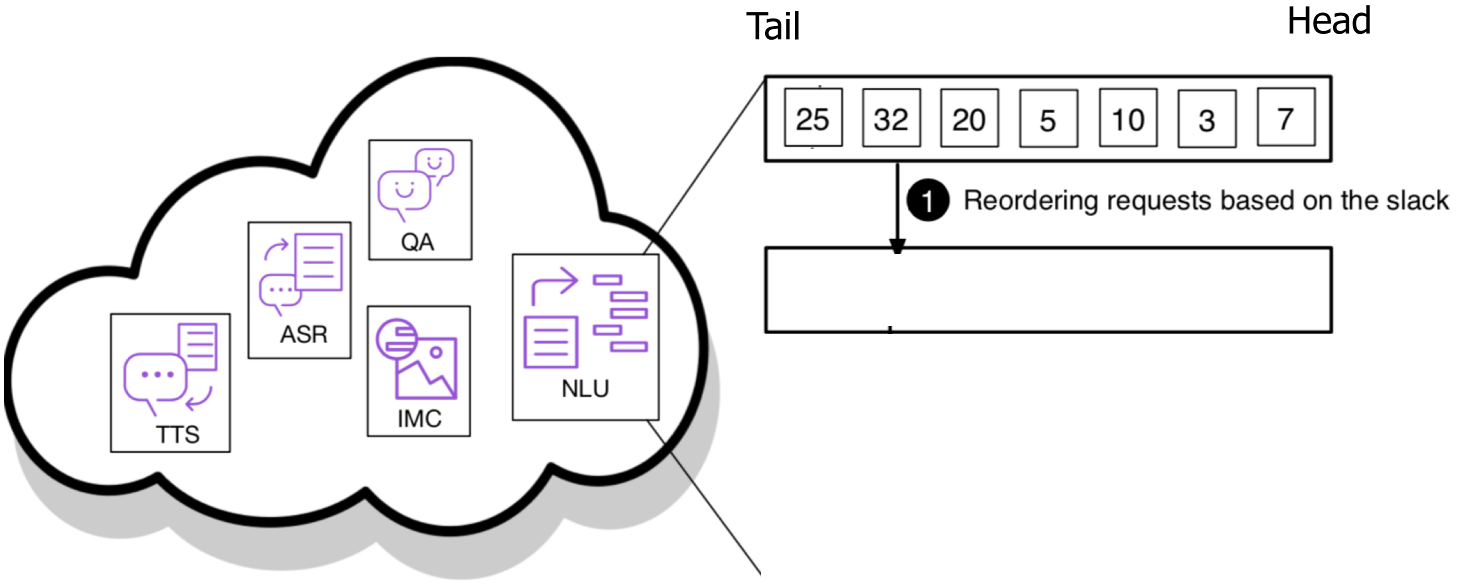
App: Pose Estimation for Sign Language (4 microservices)

1. Computation time across stages vary by a lot.

2. Percentage of slack does not vary much across batch sizes.

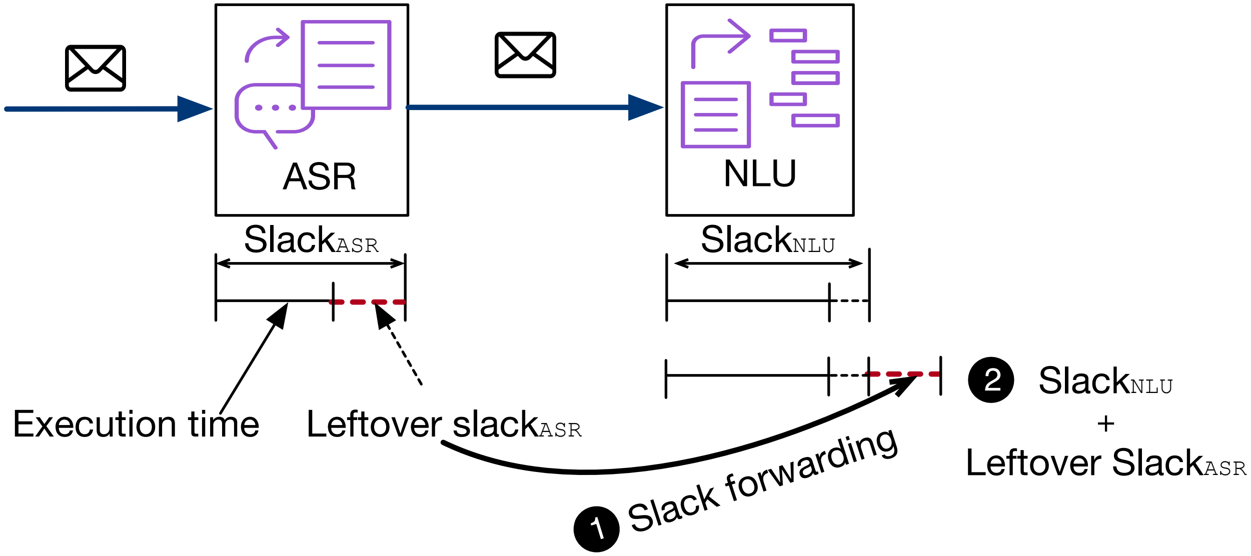
Stage Slack based Request Handling

- Prioritizing the execution with lower slack
- Dynamically batching requests based on slack



Slack Forwarding

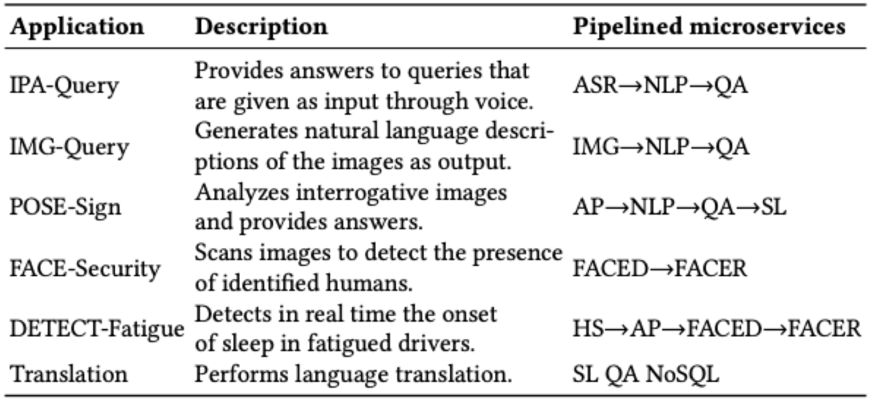
- Unused slack can be utilized later



- It can increase the overall request slack in the later stages of execution
  - Lead to enabling higher sharing degrees

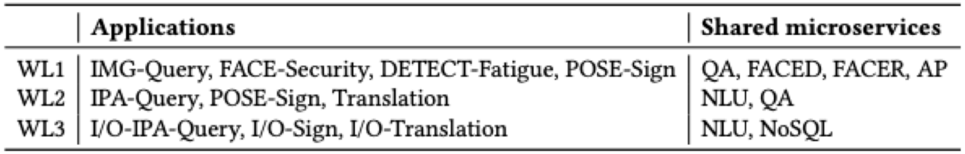
Evaluation

- Experimental platforms
  - CPU: Intel Xeon E5-2630, E3-1420
  - GPU: Nvidia GTX Titan X, GTX 1080
  - Each microservice run on a docker container
- Applications used (implemented on TensorFlow)



Application	Description	Pipelined microservices
IPA-Query	Provides answers to queries that are given as input through voice.	ASR→NLP→QA
IMG-Query	Generates natural language descriptions of the images as output.	IMG→NLP→QA
POSE-Sign	Analyzes interrogative images and provides answers.	AP→NLP→QA→SL
FACE-Security	Scans images to detect the presence of identified humans.	FACED→FACER
DETECT-Fatigue	Detects in real time the onset of sleep in fatigued drivers.	HS→AP→FACED→FACER
Translation	Performs language translation.	SL QA NoSQL

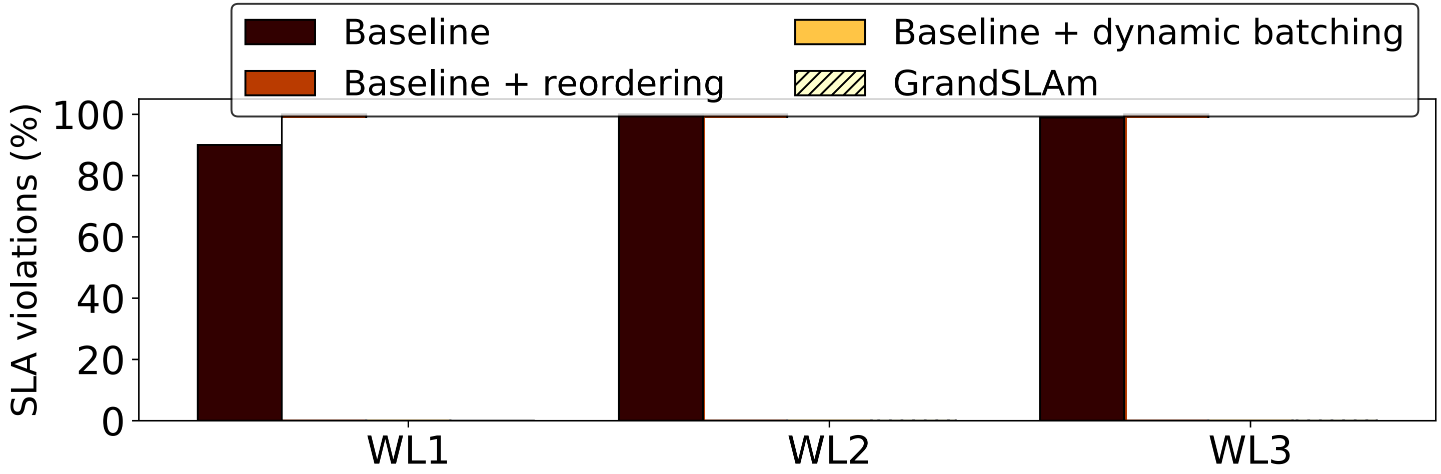
- Three workload scenarios



	Applications	Shared microservices
WL1	IMG-Query, FACE-Security, DETECT-Fatigue, POSE-Sign	QA, FACED, FACER, AP
WL2	IPA-Query, POSE-Sign, Translation	NLU, QA
WL3	I/O-IPA-Query, I/O-Sign, I/O-Translation	NLU, NoSQL

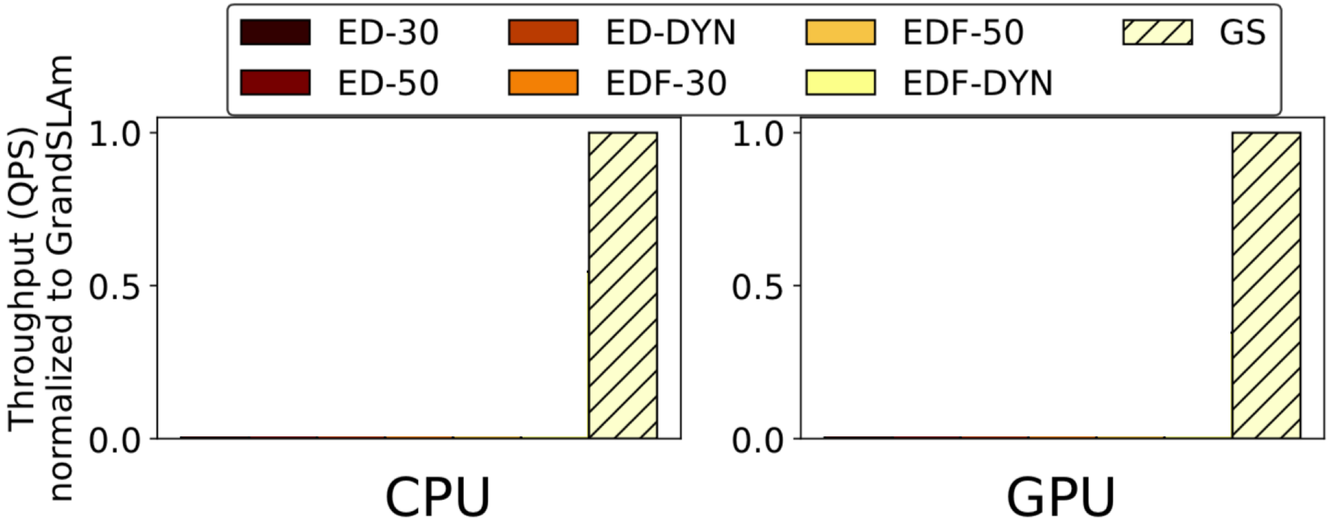
SLA: Latency Violation

- GrandSLAm improves percentage of requests that violate SLA
  - Baseline: Executes requests in a FIFO fashion without sharing the microservices



Utilization: Throughput

- ED: Equally Division
- EDF: Earliest Deadline First
- Batch size: 30, 50, DYN



Conclusions

- We explored a new approach to improve resource utilization while not violating SLAs

- Three distinct contributions
    - Analysis of microservice execution scenarios
    - Accurate estimation of completion time at each microservice
    - Guarantee end-to-end SLAs by exploiting stage level SLAs

- Future work
    - Enhancing the model to handle complex execution models
        - e.g., Parallel execution of multiple microservices, conditional execution of microservices

Thank You!

GrandSLAm: Guaranteeing SLAs for Jobs in Microservices Execution Frameworks

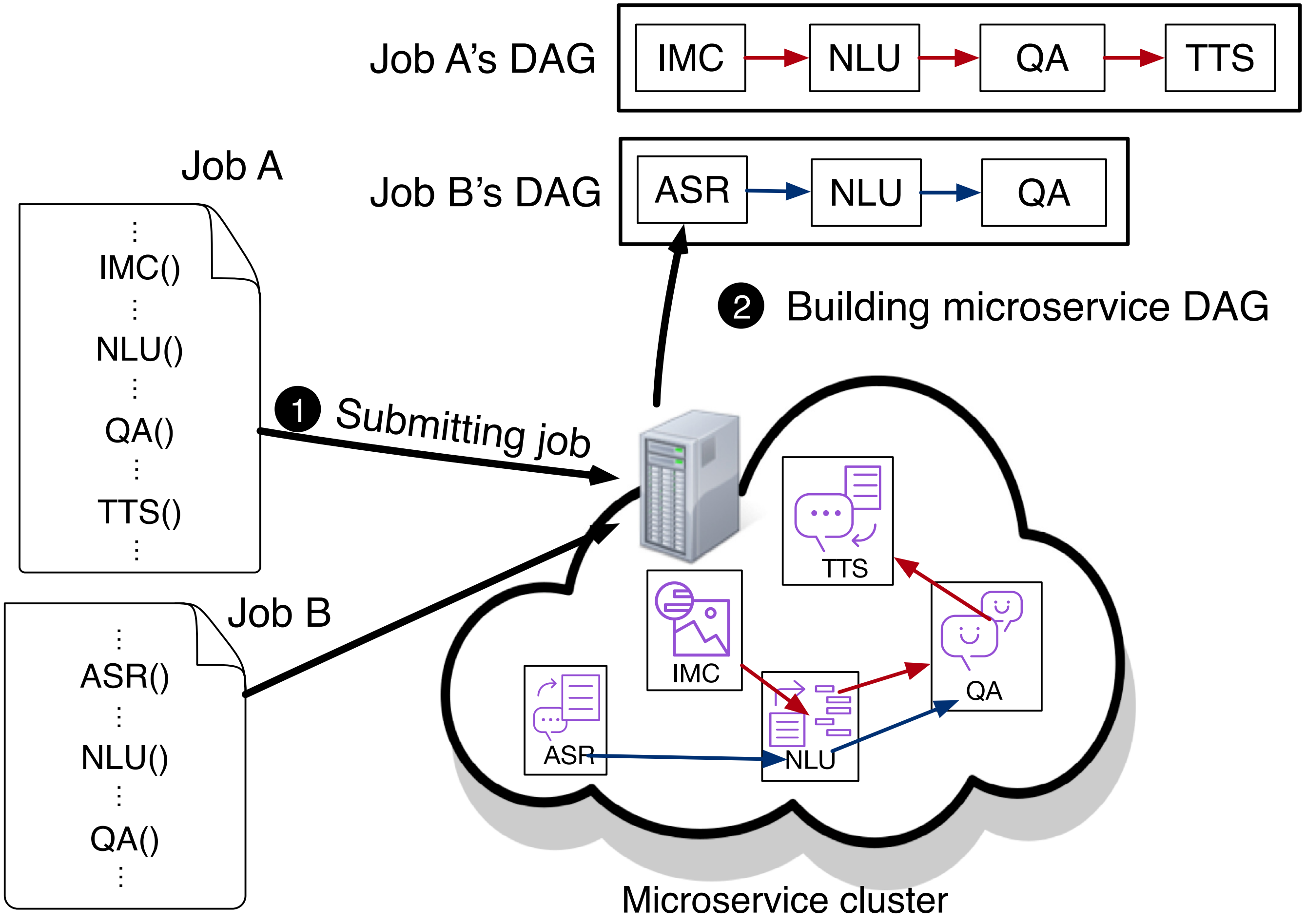
Ram Srivatsa Kannan, Lavanya Subramanian, Ashwin Raju,

Jeongseob Ahn, Jason Mars, Lingjia Tang

Expected Questions

- PLEASE LIST UP HERE

Building Microservice DAGs


Job A's DAG
IMC
NLU
QA
TTS
Job B's DAG
ASR
NLU
QA
Job A
⋮
IMC()
⋮
NLU()
⋮
QA()
⋮
TTS()
⋮
Job B
⋮
ASR()
⋮
NLU()
⋮
QA()
⋮
❶ Submitting job
❷ Building microservice DAG
TTS
IMC
QA
ASR
NLU
Microservice cluster